Qualification Accredited

OCR
Oxford Cambridge and RSA

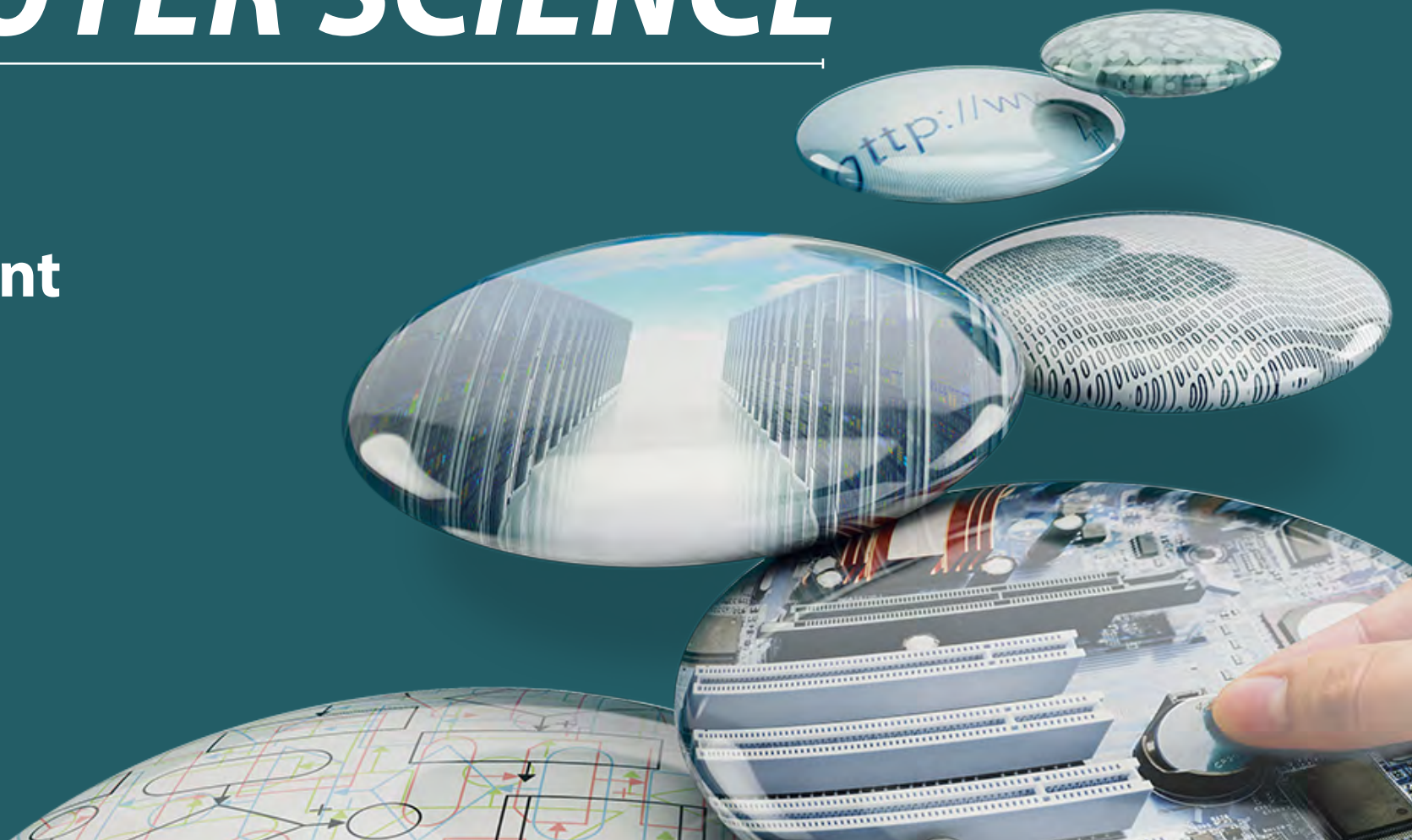
A LEVEL
Clarification Guide

COMPUTER SCIENCE

H446
For first teaching in 2015

Subject content clarification

Version 2



www.ocr.org.uk/computerscience

Contents

A LEVEL
COMPUTER SCIENCE

Component 1

Component 1

1.1. The characteristics of contemporary processors, input, output and storage devices

Components of a computer and their uses		Content clarification	Links to other topics
1.1.1 Structure and function of the processor	a) The Arithmetic and Logic Unit; ALU, Control Unit and Registers (Program Counter; PC, Accumulator; ACC, Memory Address Register; MAR, Memory Data Register; MDR, Current Instruction Register; CIR). Buses: data, address and control: How this relates to assembly language programs. b) The Fetch-Decode-Execute Cycle, including its effect on registers. c) The factors affecting the performance of the CPU: clock speed, number of cores, cache. d) The use of pipelining in a processor to improve efficiency. e) Von Neumann, Harvard and contemporary processor architecture.	Candidates need to have an understanding of the purpose and function of the core components of a processor. Candidates need to understand the role and components of the ALU. Candidates need to understand the purpose and function or registers within the processor, including the PC, accumulator, MAR, MDR and CIR. Candidates need to understand the purpose, function and role of the data, address and control buses in the processor. Candidates need to understand how assembly language makes use of registers, and how data and addressed are transferred between registers. Candidates need to understand the purpose and stages within the FDE cycle. Candidates need to understand how and when the registers are used within this cycle, and how and where data and addresses are transmitted to/from in each part of this cycle. Candidates need to understand how the performance of the CPU can be affected by many factors. Candidates need to understand how and why the performance is affected by the clock speed, the number of cores and the size and speed of the cache. Candidates need to have an understanding of the Von Neumann and Harvard architectures. They should be aware of the different approaches the architectures take to storing instructions and data in memory and the benefits of each approach. Candidates will not be asked about specific aspects of "contemporary processor architecture" unless explicitly named in the specification. They may, however, be asked to show an awareness of how contemporary processors differ from a pure Von Neumann architecture in more open questions.	**1.1.3 Input, output and storage** **1.2.4 Types of Programming Language** http://www.teach-ict.com/2016/AS_Computing/OCR_H046/1_1_characteristics_components/111_architecture/von_neumann/home_as_von_neumann.html https://youtu.be/UdHK35N-Kuo http://www.bbc.co.uk/education/guides/zmb9mp3/revision/4 https://youtu.be/OTDTdTYld2g https://youtu.be/Nz2JJXYKWJc https://youtu.be/ULUb4xwqz4A http://web.cs.iastate.edu/~prabhu/Tutorial/PIPELINE/pipe_title.html https://youtu.be/4WFzOyUNkaM

Components of a computer and their uses		Content clarification	Links to other topics
1.1.2 Types of processor	a) The differences between, and uses of, CISC and RISC processors. b) GPUs and their uses (including those not related to graphics). c) Multicore and Parallel systems.	Candidates need to understand the differences between the CISC and RISC processors and the key features and benefits of each. They should be aware of the relative benefits of each architecture. Candidates need to understand the purpose of GPUs and what applications they are used for (candidates need to understand how GPUs are used to aid graphics, but also other applications for example their use in modeling, data mining, etc.). Candidates should understand the benefits and using GPUs and why they are suited to certain tasks (specialist instructions, multiple cores and SIMD processing). Candidates need to understand what is meant by a parallel system and the benefits and limitations of parallel processing. Candidates need to understand that parallel processing can be achieved through different methods (i.e. multiple processors in the same computer or distributed across multiple cores in a CPU or GPU). Candidates need to understand the benefits of a multicore system in terms of parallel processing and running multiple programs at the same time.	**1.1.1 Structure and function of the processor** https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/ https://youtu.be/BJpMmq9gQE8 http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_3/parallel_processors/miniweb/pg5.htm https://youtu.be/kQ-xG_WVWjg https://youtu.be/CntADU-4_Gw
1.1.3 Input, output and storage	a) How different input output and storage devices can be applied as a solution of different problems. b) The uses of magnetic, flash and optical storage devices. c) RAM and ROM. d) Virtual storage.	Candidates need to have an understanding of a range of input, output and storage devices. Candidates do not need to understand how the input and output devices work, but must be able to recommend appropriate devices for specific situations and be able to justify choices made. Candidates need to understand that there are different types of storage device. They need to know about the characteristics of each type (magnetic, optical and flash) and understand the benefits and drawbacks of each, and be able to recommend an appropriate type of device for a given situation and justify the choice. Candidates need to understand the purpose of ROM and RAM within a computer system, their characteristics, and the role they play in the running of a range of different computers e.g. mobile devices, embedded systems etc. Candidates need to understand why there is a need for virtual storage, how virtual storage works and the benefits and drawbacks of using virtual storage. Virtual storage would be that which may appear to be local but is physically located elsewhere on the network/remotely/in the cloud.	**1.1.1 Structure and function of the processor** http://www.computerhope.com/issues/ch001361.htm https://youtu.be/vbFDsSnfLfw https://youtu.be/WjZoIgnayU4

1.2. Software and software development

Types of software and the different methodologies used to develop software		Content clarification	Links to other topics
1.2.1 Operating Systems	a) The need for, function and purpose of operating systems. b) Memory management (paging, segmentation and virtual memory). c) Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the fetch decode execute cycle. d) Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time. e) Distributed, embedded, multi-tasking, multi-user and real time operating systems. f) BIOS. g) Device drivers. h) Virtual machines, any instance where software is used to take on the function of a machine, including executing intermediate code or running an operating system within another.	Candidates need to have an understanding of why an operating system is required, along with the different tasks it performs within a computer system (e.g. resource management, file management, interrupt handling, security, providing a platform for software to run, providing a user interface and providing utilities). Candidates need to understand how operating systems manage memory. They need to understand the need for, purpose and function of paging to divide memory into usable fixed-size pages and how this aids in the transfer of memory for example virtual memory. Candidates need to understand what is meant by segmentation and how memory is divided into segments to allow access to memory. Candidates need to understand what is meant by virtual memory and why this is needed in a computer system. Candidates need to understand how paging is used in virtual memory, and the benefits and drawbacks of having and using virtual memory in a computer system. Candidates need to understand the purpose of interrupts within a computer system, why an interrupt might be generated and what happens within the CPU and memory in order to call an interrupt service routine. Candidates need to understand the need for scheduling of tasks by an operating system and the benefits that scheduling brings. Candidates need to understand that there are different scheduling algorithms, with each having benefits and drawbacks for tasks with specific characteristics. Candidates need to understand how the following scheduling algorithms work; round robin, first come first served, multi-level feedback queue, shortest job first and shortest remaining time. Candidates need to understand the different (and often overlapping) classifications of operating systems (distributed, embedded, multi-tasking, multi-user and real time), including the key features of each. They should be able to recommend (and justify) a type of operating system for a given scenario. Candidates need to understand the role of the BIOS in a computer system, and the steps that the BIOS goes through to start a computer. Candidates need to understand what is meant by 'device drivers' and why they are needed for communication between hardware and the operating system. Candidates should be able to describe what is meant by a virtual machine, how they can be used to execute intermediate code, how they can be used to run a software driven machine inside a physical machine and the benefits and drawbacks of each approach.	**1.1.1 Structure and function of the processor** **1.1.2 Types of processor** **1.1.3 Input, output and storage** **1.2.2 Applications Generation** http://www.teach-ict.com/2016/AS_Computing/OCR_H046/1_2_software/121_operating_systems/purpose_of_os/home_os_purpose.html https://youtu.be/e9klVeFgzMI http://www.studytonight.com/operating-system/cpu-scheduling https://www.tutorialspoint.com/operating_system/os_types.htm http://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading07.htm

Types of software and the different methodologies used to develop software		Content clarification	Links to other topics
1.2.2 Applications generation	a) The nature of applications, justifying suitable applications for a specific purpose. b) Utilities. c) Open source vs closed source. d) Translators: interpreters, compilers and assemblers. e) Stages of compilation (Lexical analysis, Syntax analysis, Code generation and Optimisation). f) Linkers and loaders and use of libraries.	Candidates need to understand the purpose of applications, and should have knowledge and experience of a range of different application software (for example database, word processor, web browser, graphics manipulation etc.). Candidates should be able to recommend the use of specific and generic applications for given scenarios, justifying their use and function(s) for a scenario. Candidates need to understand the purpose and role of utility software in a computer system. Candidates should be familiar with a range of utility software (e.g. disk defragmentation, file management, device driver, system cleanup, security etc.) Candidates need to be able to explain the differences between open and closed source software, the benefits and drawbacks to creator and user of each of the licensing models, and be able to recommend which is used (with justification) for a specific scenario. Candidates need to understand the need for translators when writing programs. They need to have knowledge of the differences in operation of interpreters and compilers, from these they need to be able to assess the benefits and drawbacks of using each type, and recommend with justification which should be used in a specific scenario. Candidates need to understand the role of an assembler and how it differs from interpreters and compilers. Candidates need to understand that there are a number of stages involved in compilation. Candidates need to understand how lexical analysis works and how the code is converted into tokens with the removal of unnecessary elements (e.g. comments and whitespace). Candidates need to understand how syntax errors are identified and reported at the end of the syntax analysis. Candidates need to understand how the abstract syntax tree will be fed into the next stage of code generation, and that the object code is then created. Candidates need to understand why optimisation is important and how the results of lexical analysis feeds into syntax analysis, and how the tokens are checked to ensure they meet during (and after) code generation. Candidates need to understand what code libraries are, how they are used and the benefits and drawbacks from using libraries. Candidates should have experience of using libraries to write programs. Candidates should understand how libraries are used during compilation, and how linkers and loaders are used to combine the code and library code into the final executable file.	**2.2.1 Programming techniques** https://youtu.be/gIzbuiSnuUA https://youtu.be/kPp_XPbr9Cw https://www.itgct.com/whats-the-difference-between-open-source-and-closed-source-software/ https://youtu.be/sjBR-YL828o https://www.tutorialspoint.com/compiler_design/compiler_design_lexical_analysis.htm https://www.tutorialspoint.com/compiler_design/compiler_design_syntax_analysis.htm https://www.tutorialspoint.com/compiler_design/compiler_design_code_generation.htm https://www.tutorialspoint.com/compiler_design/compiler_design_code_optimization.htm

Types of software and the different methodologies used to develop software		Content clarification	Links to other topics
1.2.3 Software Development	a) Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development. b) The relative merits and drawbacks of different methodologies and when they might be used. c) Writing and following algorithms.	Candidates need to understand the following models that can be followed to produce a software system; the waterfall lifecycle, agile methodologies (specifically extreme programming); the spiral model and rapid application development). Candidates need to understand the tasks, processes, benefits and drawbacks of each model and the similarities and differences between each. They need to understand where each model is most suitable to use, and be able to justify the use in a situation. Candidates need to be able to write algorithms using pseudocode and/or program code. Candidates need to be able to follow the code as shown in the OCR pseudocode guide, but are not expected to write code in this syntax. Candidate's code is not expected to be syntactically correct, but must use appropriate code structures.	https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm http://agilemethodology.org/ https://www.tutorialspoint.com/sdlc/sdlc_rad_model.htm https://narbit.wordpress.com/2012/06/10/the-differences-between-life-cycle-models-advantages-and-disadvantages/
1.2.4 Types of Programming Language	a) Need for and characteristics of a variety of programming paradigms. b) Procedural languages. c) Assembly language (including following and writing simple programs with the Little Man Computer instruction set). See appendix 5d. d) Modes of addressing memory (immediate, direct, indirect and indexed). e) Object-oriented languages (See appendix 5d for pseudocode style) with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism.	Candidates need to understand that there are a variety of types of programming paradigms such as procedural, OOP, low-level, and that each has its strengths and weaknesses in specific scenarios, topics or areas. Candidates need to have knowledge and experience of using a procedural programming language for example Python, VB.NET etc. Candidates need to be experienced in using procedural programming features such as (but not limited to) variables, constants, selection, iteration, sequence, subroutines, string handling, file handling, Boolean and arithmetic operators. Candidates need to be able to read, trace, amend and write procedural program code. Candidates need to have an understanding of the purpose and need for assembly language. They need to be familiar with the instructions given in Appendix 5d. They should be able to read, write, trace and amend programs written in the Little Man Computer language. Candidates need an understanding of addressing, which should be integrated with assembly language. Candidates should have experience of using immediate, direct, indirect and indexed addressing in the writing, reading and tracing of programs written in assembly language. Candidates need to understand object-oriented code (as specified in the pseudocode guide). They need to have an understanding of classes, objects, attributes and methods. They need to understand the difference between private and public attributes and methods. Candidates need to understand encapsulation and the use of get and set methods to access private attributes. Candidates need to understand the purpose and principles of inheritance. Candidates need to have an understanding of polymorphism and how it can be used within a program. Candidates need to be able to read, trace, amend and write code that makes use of these object-oriented techniques.	**2.2.1 Programming techniques** **2.3.1 Algorithms** https://www.techopedia.com/definition/8982/procedural-language https://www.youtube.com/watch?v=v2FkplTqRI4 http://peterhigginson.co.uk/lmc/ http://www.yorku.ca/sychen/research/LMC/ https://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep https://youtu.be/5d7KwRWZUiA https://youtu.be/0s3WzPt9ni8 https://youtu.be/wIbDS2_nFGg https://youtu.be/dk96fq2W-S0 https://youtu.be/E1Vh-z4HjWw https://youtu.be/yYqE5JKqLbw http://www.teach- ict.com/as_as_computing/ocr/H447/F453/3_3_8/lowlevel/miniweb/index.htm

1.3. Exchanging data

How data is exchanged between different systems		Content clarification	Links to other topics
1.3.1 Compression, Encryption and Hashing	a) Lossy vs Lossless compression. b) Run length encoding and dictionary coding for lossless compression. c) Symmetric and asymmetric encryption. d) Different uses of hashing.	Candidates need to understand the need for compression, especially when transferring data via the Internet. Candidates need to understand the difference between lossy and lossless compression, and the benefits and drawbacks of each type. Candidates need to be able to recommend a type of compression for a given scenario. Candidates need to understand how run-length encoding can reduce the size of a file for example with a text file or image. Candidates should understand how dictionary coding works by substituting entries with a unique code. Candidates should have practical experience of using these algorithms with small example files. Candidates should understand the need for encryption. Candidates should understand how symmetric and asymmetric encryption work to encrypt and decrypt data. Candidates should understand the need for and purpose of using hashing algorithms to store data. Candidates should be aware of different uses for hashing, such as the storing of passwords.	**1.4.2 Data structures** https://youtu.be/ql1udIvsYmw https://youtu.be/X7TNCv-ysn4 http://computer.howstuffworks.com/encryption2.htm https://youtu.be/6Arv1wvAZVo https://youtu.be/nXuVr7NyV7s

How data is exchanged between different systems		Content clarification	Links to other topics
1.3.2 Databases	a) Relational database, flat file, primary key, foreign key, secondary key, entity relationship modeling, normalisation and indexing. See appendix 5g. b) Methods of capturing, selecting, managing and exchanging data. c) Normalisation to 3NF. d) SQL - Interpret and modify. See appendix 5d. e) Referential Integrity. f) Transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy.	Candidates need to understand what is meant by a database. Candidates should be familiar with basic database terminology such as fields, records and tables. Candidates should know the difference between a flat file and a relational database, and be able to explain the benefits and limitations of each approach. Candidates should be able to use Normalisation to produce a suitable database schema and explain the benefits of doing so. They should be able to normalise up to 3NF. Candidates should know and be able to apply the criteria for 1NF, 2NF and 3NF. Candidates should know what is meant by a primary key, foreign key and secondary key and how each are used in a database. Candidates should be able produce and follow Entity relationship diagrams which include 1:1, 1:M and M:M relationships. They should be able to identify how tables should be linked. Candidates need to have an awareness of a range of methods for capturing data (such as forms, OCR, OMR and sensors) selecting data (such as Query By Example and SQL), managing data (such as changing data by manipulating it – e.g. arithmetic functions, adding, editing, deleting the data) and exchanging data (with commons formats such as CSV and JSON). Candidates won't be specifically asked about any one of these methods but may be asked to discuss/justify suitable methods as part of a more open question. Candidates need to have an understanding of the need to interrogate data within a database. Candidates should understand the purpose of indexing in a database and the benefits of using indexing to optimise the searching for data. Candidates need to have experience of a range of methods for capturing data (such as forms – what do they collect, what do they look like - data mining, where does the data come from, how is it collected and analysed), selecting data (such as how to produce QBEs – adding fields, tables, criteria, sorting - selecting through Boolean expressions – AND, OR, NOT), managing data (such as changing data by manipulating it – e.g. arithmetic functions - , adding, editing, deleting the data) and exchanging data (such as methods of transferring data – electronic i.e. memory stick, e-mail, and non-electronic e.g. paper based - appropriate formats for the transfer of data and communication mediums to transfer data – such as the structure, is it in a table or a list). Candidates should have experience of using SQL to edit and modify data in a database. They should understand the need for SQL as a standard language. Candidates should be able write and follow scripts using the SQL commands listed in appendix 5d. Candidates need to understand what is meant by referential integrity, and why this is desirable in a database. Candidates should understand what is meant by transaction processing, and scenarios where transaction processing takes place. Candidates should understand the problems that arise from transaction processing, and how these can be overcome. Candidates should understand the ACID rules for transaction processing, and why databases should be built to these standards. Candidates should understand how record locking prevents the overriding of data, and understand how record locking takes place.	http://www.studytonight.com/dbms/database-normalization.php https://www.lucidchart.com/pages/er-diagrams https://youtu.be/ob7Zy8NgVK4 http://www.w3schools.com/sql/ http://searchsqlserver.techtarget.com/definition/ACID http://www.service-architecture.com/articles/database/acid_properties.html

How data is exchanged between different systems		Content clarification	Links to other topics
1.3.3 Networks	a) Characteristics of networks and the importance of protocols and standards. b) The internet structure: • The TCP/IP Stack. • DNS • Protocol layering. • LANs and WANs. • Packet and circuit switching. c) Network security and threats, use of firewalls, proxies and encryption. d) Network hardware. e) Client-server and peer to peer.	Candidates need to understand the definition and purpose of a network. Candidates need to understand the purpose of, and importance of using protocols. They should be able to discuss examples of protocols that may be used in a network/the internet (but will not be asked to recall information about any specific protocol). Candidates should understand the term standard and the purpose and need for standards in a network (or any situation where data is transferred). Candidates need to understand the purpose and benefits of layering protocols, particularly within the TCP/IP stack. Candidates need to know the different layers within the TCP/IP stack and the purpose of each. Candidates need to understand how data is transmitted on the Internet, the use of IP addresses and packets in the transfer of data. (NB Candidates are not expected to be familiar with the OSI model). Candidates are expected to understand the terms LAN and WAN. Candidates need to understand how the Domain Name System is used to find the IP of a URL is found. Candidates need to understand the purpose, function, benefits and drawbacks of both packet and circuit switching. Candidates need to be able to suggest packet or circuit switching for a given scenario. Candidates need to understand that there are a range of security issues and threats involved with networked computers. Candidates need to be aware of threats such as hackers, viruses, unauthorised access, denial of service, spyware, SQL injection, phishing and pharming. Candidates need to know about ways of minimising, or preventing these threats for example firewalls, secure passwords, anti-virus, anti-spyware etc. Candidates need to have knowledge of the hardware required to connect to and/or build a network (e.g. modem, router, cable, NIC, Wireless Access Points, hub, switch etc). Candidates need to understand the purpose of the hardware, but are not required to understand how they physically work. Candidates need to understand the difference between a client-server and peer-to-peer network. Candidates need to know the benefits and drawbacks of each type of network and be able to recommend one for a given scenario.	**1.3.4 Web Technologies** https://youtu.be/1gdrZwBouOs http://www.computerworld.com/article/2593382/networking/networking-packet-switched-vs-circuit-switched-networks.html https://www.getcybersafe.gc.ca/cnt/rsks/cmmn-thrts-en.aspx

How data is exchanged between different systems		Content clarification	Links to other topics
1.3.4 Web Technologies	a) HTML, CSS and JavaScript. See appendix 5d. b) Search engine indexing. c) PageRank algorithm. d) Server and client side processing.	Candidates need to understand the purpose of HTML, CSS and JavaScript. Candidates should have experience of writing webpages using HTML, CSS and JavaScript. Candidates need to be able to recognise the code in Appendix 5d, and be able to read, write, amend and interpret code using HTML, CSS and JavaScript. Candidates should understand how and why search engine results are indexed. They should understand how PageRank ranks these results. Candidates should understand how page rank works at a high level but are not expected to be able to code the algorithm. Candidates need to understand the difference between server and client side processing, and should be aware of examples (for example Javascript code vs PHP code) of processing on both sides. Candidates should be aware of the benefits and drawbacks of both types of processing.	**1.3.3 Networks** http://www.w3schools.com/html/ http://www.w3schools.com/js/default.asp http://www.w3schools.com/css/default.asp https://youtu.be/faJFlcVJIDM http://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm https://youtu.be/715N3qyrYJk http://www.htmlbasictutor.ca/search-engine-indexing.htm

1.4. Data types, data structures and algorithms

How data is represented and stored within different structures. Different algorithms that can be applied to these structures		Content clarification	Links to other topics
1.4.1 Data Types	a) Primitive data types, integer, real/ floating point, character, string and Boolean. b) Represent positive integers in binary. c) Use of sign and magnitude and two's complement to represent negative numbers in binary. d) Addition and subtraction of binary integers. e) Represent positive integers in hexadecimal. f) Convert positive integers between Binary Hexadecimal and denary. g) Representation and normalisation of floating point numbers in binary. h) Floating point arithmetic, positive and negative numbers, addition and subtraction. i) Bitwise manipulation and masks: shifts, combining with AND, OR, and XOR. j) How character sets (ASCII and UNICODE) are used to represent text.	Candidates need to have an understanding of programming data types such as integer, real, Boolean, character string etc. Candidates need to be able to choose appropriate data types for a situation or given data. Candidates should have experience of programming solutions using these data types. Candidates should have knowledge of how to convert from one data type to another. Candidates should understand how and why computers store data as binary using switches, and that a binary number can have a variety of different interpretations depending on what is being stored (e.g. numeric, text, image, sound). Candidates should be able to convert positive whole numbers to binary and from binary to decimal/denary. Candidates should know how to store negative numbers using Sign and Magnitude and two's complement. Candidates should be able to convert denary numbers to sign and magnitude, and two's complement – and vice-versa. Candidates should be able to perform addition and subtraction on integer binary numbers. (These numbers could be positive or negative using two's complement representation.) Candidates need to have an understanding of the purpose and potential uses of Hexadecimal for example where and why they are used instead of binary and the benefits of using Hexadecimal over alternatives such as binary. Candidates should be able to convert denary numbers to hexadecimal and vice-versa and from binary to hexadecimal and vice-versa. Candidates should have an understanding of how (positive and negative) real numbers are represented in a binary floating-point representation, and should be able to convert between a denary number and a real binary number. (NB the representation used for the exam is the mantissa and exponent both represented using two's complement.) Candidates should understand the need for normalised floating point numbers. Candidates should be able to normalise a floating point number. Candidates should be able to perform addition and subtraction floating point arithmetic including addition and subtraction of both positive and negative numbers. Candidates should be able to perform right and left logical shifts. Candidates should understand the effect of right and left shifts on a binary numbers. Candidates should understand the purpose of using masks with bitwise operators, and should have experience of applying masks using AND, OR and XOR. Candidates should have an understanding of how characters are represented in a computer and in binary. Candidates should understand the need for a character set and how a computer makes use of a character set. Candidates should be aware of the ASCII and UNICODE character sets and be able to explain the differences between these and how these impact what can be stored. Candidates should be able to use a character set, or part of a character set, to translate characters into binary and vice-versa. (Candidates are not expected to memorise any values in a character set).	**1.4.3 Boolean Algebra** http://www.bbc.co.uk/education/guides/zwsbwmn/revision/7 http://thestarman.pcministry.com/asm/hexawhat.html http://www.bbc.co.uk/education/guides/zjfgjxs/revision/5 https://youtu.be/715N3qyrYJk https://youtu.be/YtMv4u-9poQ https://youtu.be/tKZsdbn8XQs https://youtu.be/ZpQsn4s7hBI https://youtu.be/-LNUXmMUMAM

Component 1

How data is represented and stored within different structures. Different algorithms that can be applied to these structures		Content clarification	Links to other topics
1.4.2 Data Structures	a) Arrays (of up to 3 dimensions), records, lists, tuples. b) The following structures to store data: linked-list, graph (directed and undirected), stack, queue, tree, binary search tree, hash table. c) How to create, traverse, add data to and remove data from the data structures mentioned above. *(This can be **either** using arrays and procedural programming or an object-oriented approach).*	Candidates should be able to describe what is meant by arrays (up to 3 dimensions), records, lists and tuples. They are expected to be able recognise when they can be used and incorporate them in their programs to store data. Candidates need to have an understanding of the behaviour of linked-lists, graphs, stacks, queues, trees, binary search trees and hash tables. Candidates need to be able be aware of how the aforementioned data structures can be implemented. We would recommend a general understanding of these principles that can be applied to a given scenario rather than trying to memorise code patterns. Candidates should have experience of implementing these structures in a variety of contexts, for example through a procedural program, through a different data structure and through an object-oriented approach. Candidates need to be able to read, trace and write code to implement features of these data structures. (Again we would recommend a general understanding backed up with practice implementing them, rather than trying to memorise code patterns).	**1.2.4 Types of programming language** **2.2.1 Programming techniques** **2.3.1 Algorithms** https://www.cs.cmu.edu/~adamchik/15-121/lectures/Stacks%20and%20Queues/Stacks%20and%20Queues.html https://www.tutorialspoint.com/data_structures_algorithms/stack_algorithm.htm https://www.tutorialspoint.com/data_structures_algorithms/dsa_queue.htm https://www.cs.bu.edu/teaching/c/stack/array/ https://youtu.be/b8s0-VLkVA0 https://youtu.be/K72XTSusEO0 https://www.cs.bu.edu/teaching/c/queue/array/types.html https://www.tutorialspoint.com/data_structures_algorithms/linked_lists_algorithm.htm

How data is represented and stored within different structures. Different algorithms that can be applied to these structures		Content clarification	Links to other topics
1.4.3 Boolean Algebra	a) Define problems using Boolean logic. See appendix 5d. b) Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions. c) Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws, distribution, association, commutation, double negation. d) Using logic gate diagrams and truth tables. See appendix 5d. e) The logic associated with D type flip flops, half and full adders.	Candidates should be familiar with AND, OR, NOT and XOR. Candidates should be familiar with the logic of each Boolean operator, and the truth tables. Candidates should be able to construct logic gate diagrams from a Boolean expression and vice-versa. Candidates should be able to construct truth tables from Boolean expressions and logic gate diagrams. Candidates should have an understanding that Boolean expressions can be simplified and should have experience of simplifying expressions using Karnaugh maps. Candidates should be able to create, complete and interpret Karnaugh maps to simplify Boolean expressions. Candidates should be aware of the given De Morgan's laws and should be able to apply these to a Boolean statement. Candidates should have experience of manipulating and simplifying Boolean statements using these rules of distribution, commutation, association and double negation. Candidates need to understand the purpose and principles of D type flip flops and how and where they are used in a computer. They should be able to recognise how they can be triggered by a clock pulse (see practice paper 2 for an example). Candidates are not expected to memorise the logic gates that make up a D-type flip flop. Candidates need to understand the purpose and function of an adder circuit, and the difference between a half and full adder. They should be able to recognise and draw the logic gates and truth tables for full and half adders.	**1.4.1 Data types** http://www.ee.surrey.ac.uk/Projects/Labview/minimisation/karnaugh.html http://www.32x8.com/ https://youtu.be/uTUV-VnKAdM https://youtu.be/hflkW-D3hrM https://youtu.be/osI68WZz_CM http://theteacher.info/index.php/computing-principles-01/1-4-data-types-structures-and-algorithms/1-4-3-boolean-algebra/2253-d-type-flip-flops https://youtu.be/KiU2PA3t8oc http://www.ee.surrey.ac.uk/Projects/Labview/boolalgebra/ https://youtu.be/x_wZjDSKHPk

Component 1

1.5 Legal, moral, cultural and ethical issues

The individual (moral), social (ethical) and cultural opportunities and risks of digital technology. Legislation surrounding the use of computers and ethical issues that can or may in the future arise from the use of computers		Content clarification	Links to other topics
1.5.1 Computing related legislation	a) Data Protection Act 1998. b) Computer Misuse Act 1990. c) Copyright and Patents Act 1988. d) Regulation of Investigatory Powers Act 2000.	Candidates need to have an understanding of the need for and purpose of laws relating to the use of computers. Candidates should be familiar with the purpose and role of the Data Protection Act, candidates will need to understand the different rules that are within the DPA and how these impact the use of computers and the storage of data by organisations, this should include what organisations can and cannot do. Candidates need to understand the purpose and principles of the Computer Misuse Act, including the actions that it prohibits. Candidates need to understand the purpose and principles of the Copyright and Patents Act, including the actions that it prohibits. Candidates need to understand the purpose and principles of the Regulation of Investigatory Powers Act, and what this allows in the interception and monitoring of electronic communication. Candidates need to understand how the regulations impact organisations and the use of computers and electronic communication. We are aware the law is constantly changing and some of the mentioned laws (most notably the DPA) are likely to change over the course of the specification. Answers will be accepted that use an interpretation of the law based on when the specification was started or when the examination was sat.	http://www.legislation.gov.uk/ukpga/2000/23/contents https://youtu.be/R1ymVnk5XZA https://youtu.be/DdZWxllYKQk

The individual (moral), social (ethical) and cultural opportunities and risks of digital technology. Legislation surrounding the use of computers and ethical issues that can or may in the future arise from the use of computers		Content clarification	Links to other topics
1.5.2 Ethical, moral and cultural issues	a) The individual (moral), social (ethical) and cultural opportunities and risks of digital technology: • Computers in the workforce • Automated decision making • Artificial intelligence • Environmental effects • Censorship and the Internet • Monitor behavior • Analyse personal information • Piracy and offensive communications • Layout, colour paradigms and character sets.	In order to prepare for this section we would recommend candidates regularly keep abreast of technological developments in the news. Candidates need to understand what is meant by moral, social, ethical and cultural issues in relation to the use of computers. Candidates need to understand how the use of computers, and the increasing use of computers in the work force has moral, social, ethical and cultural implications and risks to a variety of people such as the employees, employers, society and organisations. Candidates need to understand how the use of computers to make decisions automatically has moral, social, ethical and cultural implications and risks to a variety of people such as those people who make the decisions, the people the decisions affect, and the need for additional collection of information to ensure the decisions are accurate and valid. Candidates need to understand how the development of artificial intelligence has moral, social, ethical and cultural impacts on a variety of people. Candidates need to understand how the environmental effects of computers (such as disposal, energy use) have moral, social, ethical and cultural implications. Candidates need to understand how the Internet and censorship on the Internet has moral, social, ethical and cultural implications. Candidates need to understand the moral, social, ethical and cultural implications of using computers to monitor behaviour (such as CCTV, tracking phone calls, GPS, monitoring emails). Candidates need to understand the moral, social, ethical and cultural implications of using computers to analyse personal information (such as the gathering, storing and analysing of medical records) Candidates need to understand how different cultures impact on the use of and creation of computers and programs. For example languages make use of different characters, and how this in turn impacts the use of character sets. Some languages read left to right, and others right to left. Candidates should understand how colours have different meanings in different cultures for example red means danger in one culture, and luck in another. Candidates need to consider how these will impact the creation of computer applications.	http://www.bbc.co.uk/ethics/introduction/intro_1.shtml https://aitopics.org/search?filters=taxnodes:Technology%7CInformation%20Technology%7CArtificial%20Intelligence%7CIssues%7CSocial%20%26%20Ethical%20Issues https://philosophynow.org/issues/110/Surveillance_Ethics http://www.informationisbeautiful.net/visualizations/colours-in-cultures/ http://www.empower-yourself-with-color-psychology.com/cultural-color.html

Component 2

Component 2

2.1. Elements of computational thinking

Understand what is meant by computational thinking		Content clarification	Links to other topics
2.1.1 Thinking abstractly	a) The nature of abstraction. b) The need for abstraction. c) The differences between an abstraction and reality. d) Devise an abstract model for a variety of situations.	Candidates need to understand the term abstraction and its purpose in the design and creation of computer programs. Candidates need to understand the benefits of abstraction and apply these benefits to a specific scenario. Candidates may be given a scenario and be asked how abstraction can be applied to it, how it has been applied or how further abstraction can be applied. Candidates need to have an understanding of how reality differs to abstraction and understand the differences between reality and abstraction. Candidates may be given a scenario/example and be asked how the abstraction differs from the reality.	**2.1.2 Thinking ahead** **2.1.3 Thinking procedurally** **2.1.4 Thinking logically** **2.1.5 Thinking concurrently** http://www.bbc.co.uk/education/guides/zttrcdm/revision http://whatis.techtarget.com/definition/abstraction
2.1.2 Thinking ahead	a) Identify the inputs and outputs for a given situation. b) Determine the preconditions for devising a solution to a problem. c) The nature, benefits and drawbacks of caching. d) The need for reusable program components.	Candidates need to understand that situations/programs require input and output, and that the outputs can be in digital and/or hard copy format. Candidates may be given a description, diagram, or pseudocode for a scenario, and they will need to demonstrate an understanding of the inputs and outputs required and/or are used in that scenario. For a description (or similar) of a program, candidates need to be able to determine what additional conditions they need to know before they can produce a solution, and how this information may affect the solution. Candidates need to have an understanding of the purpose of caching in programming, and how it can be used when writing a program. Candidates need to be able to apply their knowledge of caching to a scenario to demonstrate an understanding of how it can be used. Candidates need to understand the benefits and drawbacks of using caching in a program.	**2.1.1 Thinking abstractly** **2.1.3 Thinking procedurally** **2.1.4 Thinking logically** **2.1.5 Thinking concurrently** https://youtu.be/5dPDrjJFUv8 https://youtu.be/Al64CX4pufY https://youtu.be/V3Hhf544-XU

Understand what is meant by computational thinking		Content clarification	Links to other topics
2.1.3 Thinking procedurally	a) Identify the components of a problem. b) Identify the components of a solution to a problem. c) Determine the order of the steps needed to solve a problem. d) Identify sub-procedures necessary to solve a problem.	Candidates need to be able to deconstruct a program and identify its component parts, for example listing the parts or completing a structure chart. Candidates may be given some component parts and be asked to add to, or complete these from a written description of pseudocode code for a program. Candidates need to be able to identify the steps that will take place to complete an algorithm, or program, and be able to write these in a suitable format (such as a flowchart or pseudocode), and put a given list into the correct order to produce a working program. Candidates may need to write pseudocode, code, or draw a flowchart to show a sequence of steps. Candidates need to understand the use and purpose of sub-procedures in a program. They need to be able to identify where sub-procedures may be used, and write appropriate pseudocode, code and/or flowchart(s) for these sub-procedures, making use of parameters where appropriate. Candidates may be given a structure diagram that they will need to interpret, or complete, to identify these sub-procedures.	**2.1.1 Thinking abstractly** **2.1.2 Thinking ahead** **2.1.4 Thinking logically** **2.1.5 Thinking concurrently** http://www.bbc.co.uk/guides/z8ngr82 https://youtu.be/u7WsR8iTTnI
2.1.4 Thinking logically	a) Identify the points in a solution where a decision has to be taken. b) Determine the logical conditions that affect the outcome of a decision. c) Determine how decisions affect flow through a program.	Candidates need to understand that decisions are made within programs, and they need to be able to identify where these decisions will take place within an algorithm or program. They need to understand what these decisions are, and the impact of these decisions (and the outcomes) on the algorithm/program. Candidates need to understand that there can be many different routes through a program, and understand how decisions influence these routes and outcomes.	**2.1.1 Thinking abstractly** **2.1.2 Thinking ahead** **2.1.3 Thinking procedurally** **2.1.5 Thinking concurrently** https://youtu.be/2ybo8KiU32k https://youtu.be/ZrPz-ENUhbs
2.1.5 Thinking concurrently	a) Determine the parts of a problem that can be tackled at the same time. b) Outline the benefits and tradeoffs that might result from concurrent processing in a particular situation.	Candidates need to understand what is meant by thinking concurrently. They need to be able to work out which parts of a program can be developed to take place (be processed) at the same time, and which parts are dependent on other parts. Candidates need to understand the benefits and trade offs that are brought from concurrent processing, and be able to apply these to a given scenario. For example, candidates need to understand how concurrent processing could be applied to a specific program, why it would be applied to that program, and what problems might arise from using it.	**2.1.1 Thinking abstractly** **2.1.2 Thinking ahead** **2.1.3 Thinking procedurally** **2.1.4 Thinking logically** http://searchoracle.techtarget.com/definition/concurrent-processing

2.2. Problem solving and programming

How computers can be used to solve problems and programs can be written to solve them *(Learners will benefit from being able to program in a procedure/imperative language and object oriented language.)*		Content clarification	Links to other topics
2.2.1 Programming techniques	a) Programming constructs: sequence, iteration, branching. b) Recursion, how it can be used and compares to an iterative approach. c) Global and local variables. d) Modularity, functions and procedures, parameter passing by value and by reference. e) Use of an IDE to develop/ debug a program. f) Use of object oriented techniques	Candidates need to understand the constructs of sequence, iteration and branching. They must be able to use these constructs independently of each other, and combine them to produce a solution. These include the selection statements of IF (including ELSEIF and ELSE) and select case statements. These also include both condition based iteration (e.g. while, repeat until) and count controlled iteration (e.g. FOR) – as well as how condition based iteration can be used as count controlled. Candidates need to have an understanding of the principle of recursion and the key features that produce a recursive algorithm such as a stopping condition. They need to be able to read and trace recursive functions, write recursive functions, and translate a recursive function to an iterative solution and vice-versa. Candidates need to have an understanding of the benefits and drawbacks of using both a recursive and iterative solution. Candidates need to be able to read, create and trace code (for example using a trace table) that use these three constructs. Candidates need to understand the use and need for variables in a program and must understand the difference, benefits and drawbacks of both global and local variables. Candidates must be able to recognise where local and global variables are used and the impact that these have on a program, for example the amount of memory used. Candidates need to understand how a program using global variables can be changed to use local variables – and vice-versa. Candidates need to understand what is meant by modular code, and how this can be produced using functions and procedures. Candidates need to understand the differences between functions and procedures and how each is used within a program. Candidates need to be able to read, trace and write code that uses functions and procedures. Candidates need to understand the purpose and use of parameters within a program, and how they are used in functions and procedures. Candidates need to be able to read, trace and write code that makes use of parameters. Candidates need to understand the difference between passing a parameter by value and by reference, and they need to understand the benefits and drawbacks of each – recommending which should be used for a given situation. Candidates need to be able to read, trace and write code that makes use of parameters passed both by value and by reference. Candidates should have had experience of using an IDE to produce code. Candidates need to understand how an IDE can be used to produce code, and understand the range of features and tools that are within an IDE, that can be used to help produce and debug a program. *… continues*	**2.3.1 Algorithms** **1.4.1 Data types** **1.4.2 Data structures** http://vle.moirahouse.co.uk/studentwebsites/ict/theteacherict/newalevel/cp1_2_1.htm http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html https://youtu.be/xBAMBDyDu0s https://youtu.be/tflZNOyF8yA https://youtu.be/1vcCx1ndV2Q https://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concepts https://youtu.be/ICRyTuX99DI https://youtu.be/Czpkk2fS7go https://youtu.be/KZr1VLp5dvg

		Candidates need to understand the purpose of object-oriented code. They need to have an understanding of classes, objects, properties, attributes, methods. They need to understand the difference between private and public properties, attributes and methods. Candidates need to understand encapsulation and the use of get and set methods to access private properties. Candidates need to understand the purpose and principles of inheritance, super-classes, parent-classes and sub-classes. Candidates need to have an understanding of polymorphism and how it can be applied to classes. Candidates need to be able to read, trace and write code that makes use of these object-oriented techniques. Candidates need to be able to interpret class diagrams to produce class definitions. Candidates need to be able to identify where object-oriented programming can be used in a solution, and derive an object-oriented solution for a given scenario.	https://www.cs.umd.edu/class/fall2002/cmsc214/Tutorial/recursion2.html http://www2.hawaii.edu/~tp_200/lectureNotes/recursion.htm

How computers can be used to solve problems and programs can be written to solve them *(Learners will benefit from being able to program in a procedure/imperative language and object oriented language.)*		Content clarification	Links to other topics
2.2.2 Computational methods	a) Features that make a problem solvable by computational methods. b) Problem Recognition. c) Problem Decomposition. d) Use of divide and conquer. e) Use of abstraction. f) Learners should apply their knowledge of: • backtracking • data mining • heuristics • performance modelling • pipelining • visualisation to solve problems.	Candidates need to be able to determine if a problem can be solved using computational methods, such as decomposition, abstraction, calculations, storage of data. Candidates need to be able to recognise a problem from a description of a scenario, decompose the problem and use abstraction to design a solution. Candidates need to understand how divide and conquer can be used within a task to split the task down into smaller tasks that are then tackled. Candidates also need to identify how tasks can be carried out simultaneously to produce a solution. Candidates need to understand the purpose of backtracking within an algorithm, for example when traversing a tree. Candidates need to be able to read, trace and write code that makes use of backtracking for a given scenario. Candidates need to understand what is meant by data mining, and how data mining is used in a situation. Candidates need to understand the complexities within data mining and how a program will search for and interrogate the data. Candidates need to understand what is meant by heuristics, and how they can be used within a program (for example the A* algorithm). Candidates should have some experience of programming a simple heuristic, and be able to apply their knowledge to a given scenario to explain the purpose and benefits of using heuristics in a solution. Candidates need to understand the principles, and purpose of performance modelling, and how it is used in the production of software. Candidates need to understand the principle of pipelining and how it is used within programming (for example the result from a process feeds into the next process). Candidates need to understand how visualisation can be used to create a mental model of what a program will do or work, and that from this they can plan ahead what is going to happen or what they will need to do.	**2.1.1 Thinking abstractly** **2.1.2 Thinking ahead** **2.1.3 Thinking procedurally** **2.1.4 Thinking logically** **2.1.5 Thinking concurrently** https://www.tutorialspoint.com/data_structures_algorithms/divide_and_conquer.htm http://algorithms.tutorialhorizon.com/introduction-to-backtracking-programming/ https://youtu.be/rubdV99bbyo http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

2.3 Algorithms

The use of algorithms to describe problems and standard algorithms		Content clarification	Links to other topics
2.3.1 Algorithms	a) Analysis and design of algorithms for a given situation. b) The suitability of different algorithms for a given task and data set, in terms of execution time and space. c) Measures and methods to determine the efficiency of different algorithms, Big O notation. (Constant, linear, polynomial, exponential and logarithmic complexity) d) Comparison of the complexity of algorithms. e) Algorithms for the main data structures, (Stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees). f) Standard algorithms (Bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search).	Candidates need to be able to write algorithms using flow charts, pseudocode and program code. Candidates need to be able to follow the code as shown in the OCR pseudocode guide, but are not expected to write code in syntax. Candidates' code is not expected to be syntactically perfect, but must use appropriate structures and techniques. Candidates need to understand that there are a range of possible solutions to a task, and that these algorithms may be different in respect to their execution time and the amount of memory they make use of. Candidates need to be able to compare different algorithms for a given data set and demonstrate an understanding of which is more efficient in terms of speed and/or memory. Candidates need to be able to compare the use of one or more algorithms against several different data sets, to determine how they will differ in their use of memory and speed of execution. Candidates need to understand how the efficiency of an algorithm is measured using Big O notation. Candidates need to understand the meaning of constant, linear, polynomial, exponential and logarithmic complexity. They need to be able to recognise and draw each of these complexities of using a graph and be able to read and write the notation. Candidates need to know the best and worst case complexities for the searching and sorting methods. Candidates need to understand the difference between best case, average case and worst case complexities and how and why these can differ for an algorithm. Candidates need to have an understanding of the situations where queues, stacks, trees etc. can be used and be able to recommend and justify their use In specific scenarios or programs. Candidates need to have an understanding of a stack as a dynamic data structure. Candidates need to be able to add and remove items to a stack. Candidates need to be able read, trace and write code to implement a stack structure (including adding and removing items). Candidates need to understand how a stack can be implemented using a different data structure, such as a static array. Candidates need to have an understanding of a queue as a dynamic data structure. Candidates need to be able to add and remove data to/from a queue. Candidates need to be able to read, trace and write code to implement a queue structure (including adding and removing items). Candidates need to understand how a queue can be implemented using a different data structure, such as a static array. Candidates need to have an understanding of a tree structure, both binary and multi-branch trees. Candidates need to be able to add and remove data to/from a tree. Candidates need to be able to read, trace and write code to implement a tree structure (including adding and removing items). Candidates need to understand how a tree can be implemented using a different data structure, such as a linked list. Candidates need to understand why and how trees are traversed. Candidates need to understand how a depth-first (post-order) traversal works, and be able to perform the traversal on a tree. Candidates need to be able to read, trace and write code for a post-order traversal. Candidates need to understand how a breadth-first traversal works, and be able to perform the search on a tree. Candidates need to be able to read, trace and write code for a breadth-first traversal on a tree. *… continues*	**2.2.1 Programming techniques** **2.2.2 Computational methods** **1.4.2 Data structures** http://bigocheatsheet.com/ https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/ https://www.youtube.com/watch?v=v4cd1O4zkGw https://www.youtube.com/watch?v=lyZQPjUT5B4 https://www.youtube.com/watch?v=ROalU379l3U https://www.youtube.com/watch?v=JQhciTuD3E8 https://www.cs.cmu.edu/~adamchik/15-121/lectures/Stacks%20and%20Queues/Stacks%20and%20Queues.html https://www.tutorialspoint.com/data_structures_algorithms/stack_algorithm.htm https://www.tutorialspoint.com/data_structures_algorithms/dsa_queue.htm https://www.cs.bu.edu/teaching/c/stack/array

		Candidates need to have an understanding of a linked list as a dynamic data structure. Candidates need to be able to add, remove and search for data to/from/in a linked list. Candidates need to be able to read, trace and write code to implement a linked list (including adding, removing and search for items). Candidates need to have an understanding of the need for searching and sorting algorithms. Candidates need to have an understanding of pre-conditions required to perform a specific algorithm. Candidates need to understand how a bubble sort works and be able to perform a bubble sort on a set of data. Candidates need to be able to read, trace and write code to perform a bubble sort. Candidates need to understand how a merge sort works and be able to perform a merge sort on a set of data. Candidates need to be able to read, trace and write code to perform a merge sort. Candidate need to understand how a quick sort works and be able to perform a quick sort on a set of data. Candidates need to be able to read, trace and write code to perform a quick sort. Candidates need to understand how Dijkstra's shortest path algorithm works. Candidates need to be able to calculate the shortest path in a graph or tree using Dijkstra's shortest path algorithm. Candidates need to be able to read and trace code that performs Dijkstra's shortest path algorithm. Candidates need to understand how the A* algorithm works. Candidates need to be able to calculate the shortest path in a graph or tree using the A* algorithm. Candidates need to be able to read and trace code that performs the A* algorithm. Candidates need to understand how a binary search works and be able to perform a binary search on a set of data. Candidates need to be able to read, trace and write code to perform a binary search. Candidates need to understand how a linear search works and be able to perform a linear search on a set of data. Candidates need to be able to read, trace and write code to perform a linear search.	https://www.youtube.com/watch?v=sFVxsglODoo https://www.youtube.com/watch?v=okr-XE8yTO8 https://www.cs.bu.edu/teaching/c/queue/array/types.html https://www.tutorialspoint.com/data_structures_algorithms/linked_lists_algorithm.htm https://www.tutorialspoint.com/graph_theory/ https://www.tutorialspoint.com/data_structures_algorithms/binary_search_tree.htm https://www.tutorialspoint.com/data_structures_algorithms/hash_data_structure.htm https://pythonschool.net/data-structures-algorithms/implementing-the-stack-class-using-oop/ https://www.youtube.com/watch?v=8Ls1RqHCOPw https://www.youtube.com/watch?v=XaqR3G_NVoo

Did you know?

OCR Resources: *the small print*

OCR's resources are provided to support the delivery of OCR qualifications, but in no way constitute an endorsed teaching method that is required by OCR. Whilst every effort is made to ensure the accuracy of the content, OCR cannot be held responsible for any errors or omissions within these resources. We update our resources on a regular basis, so please check the OCR website to ensure you have the most up to date version.

This resource may be freely copied and distributed, as long as the OCR logo and this small print remain intact and OCR is acknowledged as the originator of this work.

Our documents are updated over time. Whilst every effort is made to check all documents, there may be contradictions between published support and the specification, therefore please use the information on the latest specification at all times. Where changes are made to specifications these will be indicated within the document, there will be a new version number indicated, and a summary of the changes. If you do notice a discrepancy between the specification and a resource please contact us at: resources.feedback@ocr.org.uk.

OCR acknowledges the use of the following content: N/A

Whether you already offer OCR qualifications, are new to OCR, or are considering switching from your current provider/awarding organisation, you can request more information by completing the Expression of Interest form which can be found here: www.ocr.org.uk/expression-of-interest

Please get in touch if you want to discuss the accessibility of resources we offer to support delivery of our qualifications: resources.feedback@ocr.org.uk

Looking for a resource?

There is now a quick and easy search tool to help find **free** resources for your qualification:

www.ocr.org.uk/i-want-to/find-resources/

Need to get in touch?

If you ever have any questions about OCR qualifications or services (including administration, logistics and teaching) please feel free to get in touch with our **Customer Support Centre**.

General qualifications

Telephone 01223 553998
Facsimile 01223 552627
Email general.qualifications@ocr.org.uk

www.ocr.org.uk

OCR is part of Cambridge Assessment, a department of the University of Cambridge. *For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored.*

Cambridge Assessment

CERTIFIED ISO 9001 UKAS MANAGEMENT SYSTEMS 001